Controls Lab 1

Austin Glaser

Jay Greco

0.0.1: Step Response

We examine the step response of the system shown below:

$$G = \frac{K}{s^2+2\zeta\omega_0 s+\omega_0^2}$$

with the following parameter values:

$$K = 1\ \zeta = 1.25\ \omega_0 = 2$$

To this, we apply a simple proportional controller with parameter Kp.

Examining the step response of the system using Matlab's rlocus() function, we see that a Kp of 2.5 is the highest we can go while maintaining zero overshoot, and a Kp of 130 gives us 50% overshoot (see figures 1a and b below).

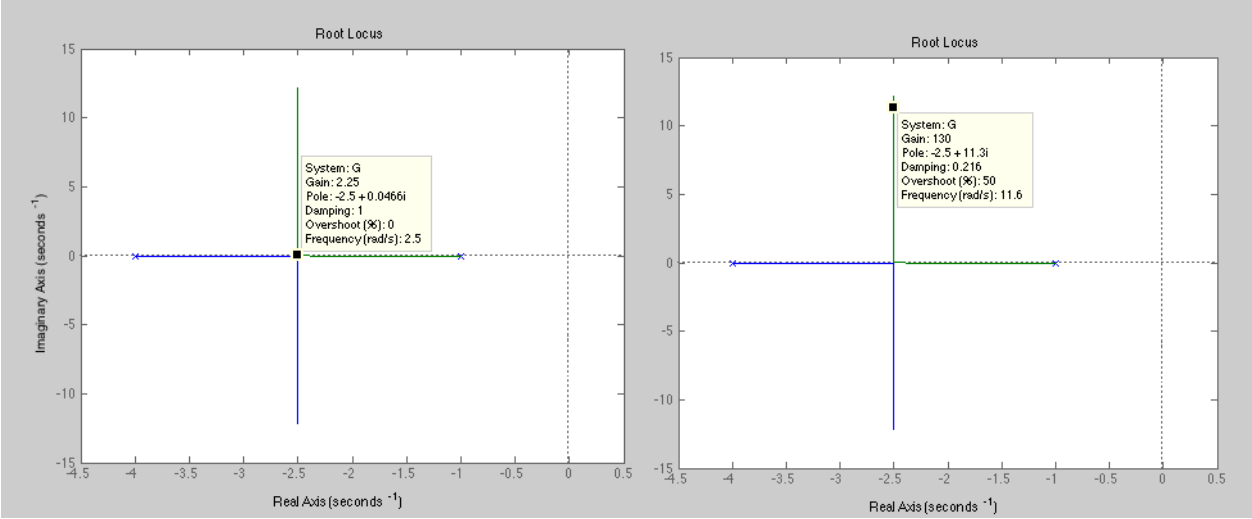


Figure 1a: Root locus plot at Kp = 2.25

Figure 1b: Root locus for Kp = 130

We then simulated the systems in labview, and exported this data into Matlab where we created plots of the time response. From these, it's immediately obvious that a higher Kp will produce a quicker rise time, but increase the controlled system's overshoot. It's less immediately obvious from the graphs, but calculating the settling time to within 2% of the final value (not the commanded value) we see that the higher Kp also produces a shorter settling time (1.47 seconds for Kp=130, as opposed to 2.33 seconds for Kp=2.5). Interestingly, our system never reaches within 2% of the setpoint of 10 (we will later find that adding an integral or proportional term will resolve this problem).

Physically, this is an extremely simple model of our system, but we can still say that its units are in degrees, or some other measure of angular position. A unit step of 10, then, corresponds to our commanding the system to move 10 degrees from an arbitrary zero point.

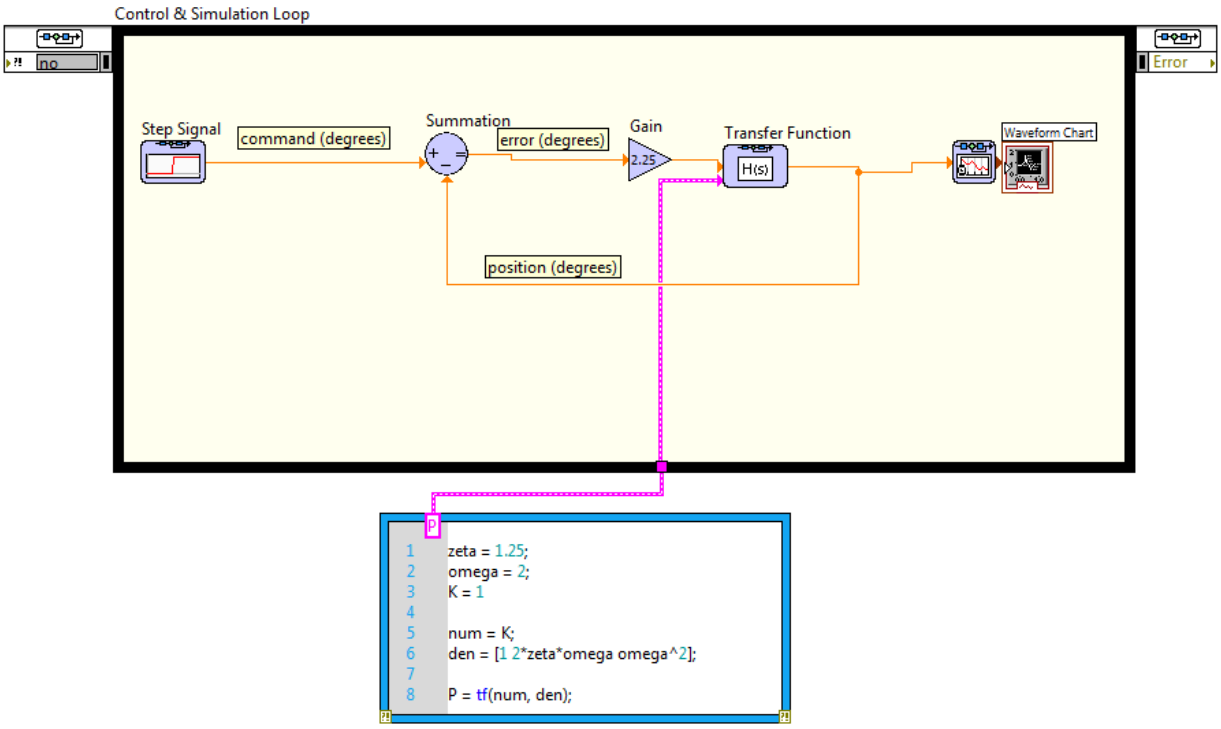


Figure 2: Labview setup for testing with units labeled

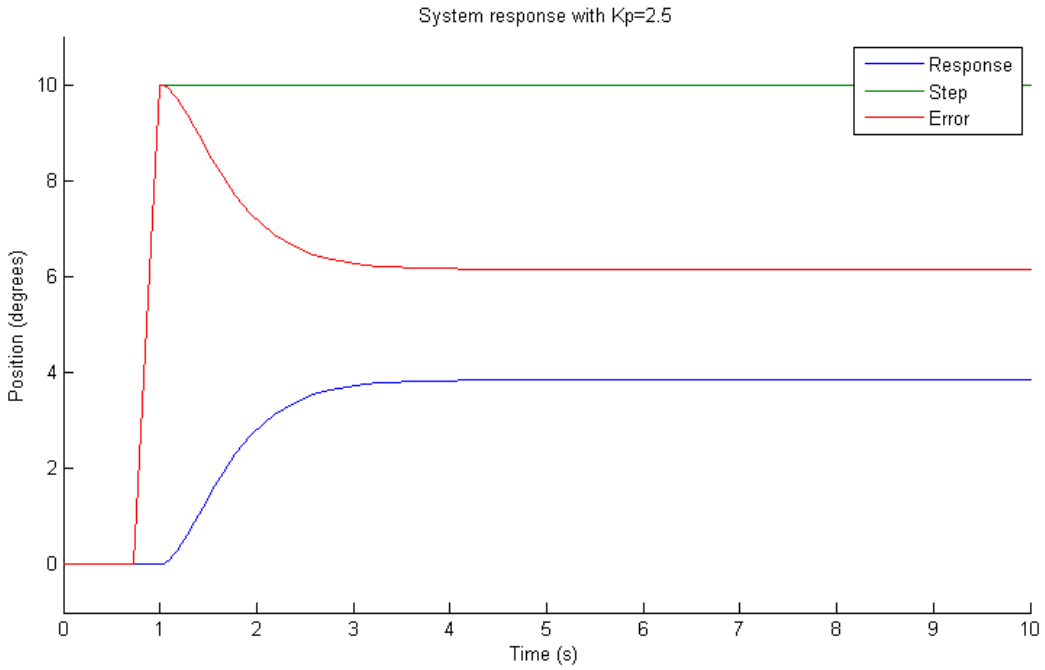


Figure 3: Step response for Kp = 2.25

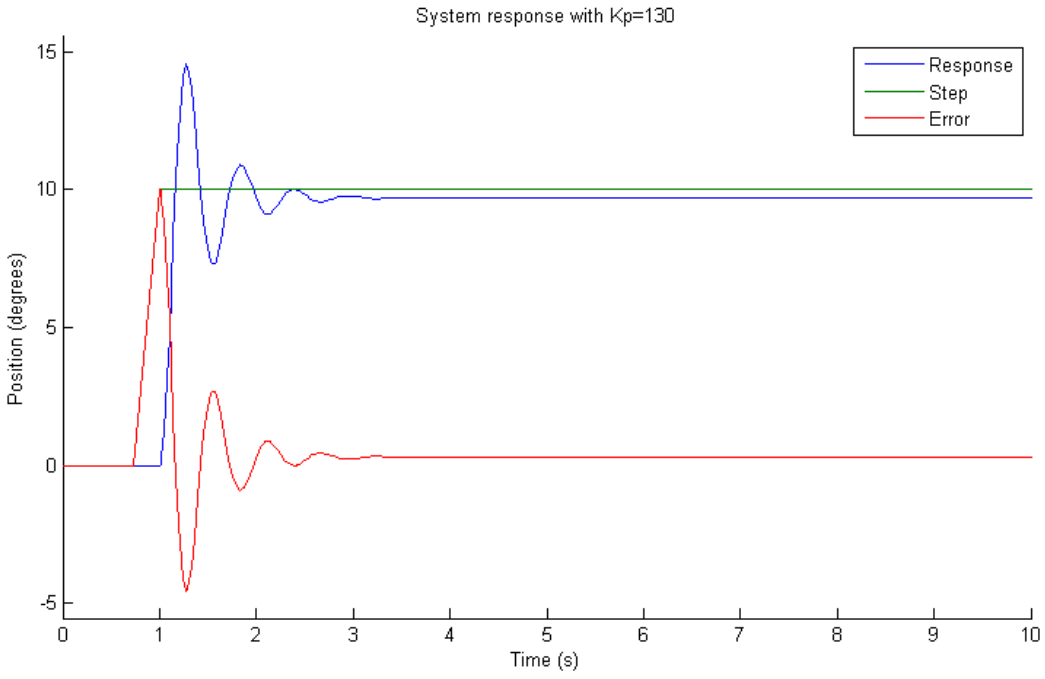
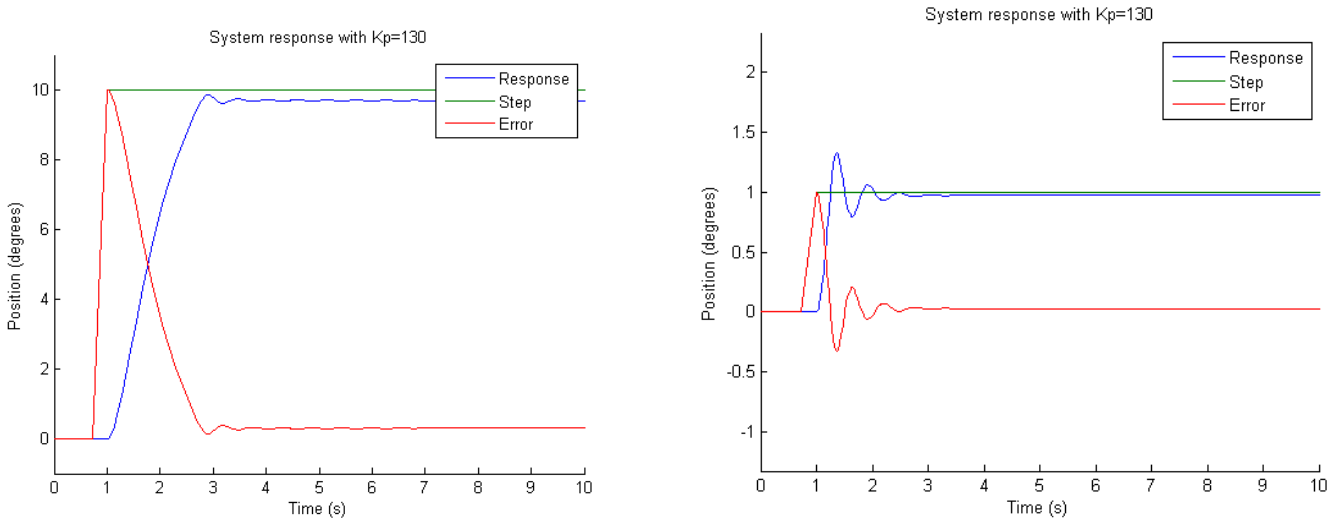


Figure 4: Step response for Kp = 130

It's clear from these plots that a higher gain will give us a faster rise time, and a lower steady-state error – though this never goes to zero.

0.1.2: Actuator Saturation

Using the LabVIEW saturation block, the control of the system was set to saturate. The saturation was set to a value of +/- 50. Using the same gains as in task 1, the 2% settling time was then recorded again. For a step of 1, the settling time increased slightly. For a step size of 10, the settling time also increased.



Figures 5a and 5b: Step response with Kp=130, simulating controller saturation. Figure 5 has a step of 10, figure 6 of 1

0.1.3: Time Delay

Using the discrete Z transform block in LabVIEW, a time delay was added into the control system. After some experimentation, it was found that a delay of 4ms or greater caused the system to go astable. This makes sense, as the controller is compensating based on what it believes to be current and accurate data, but the system is no longer in the state that was reported.

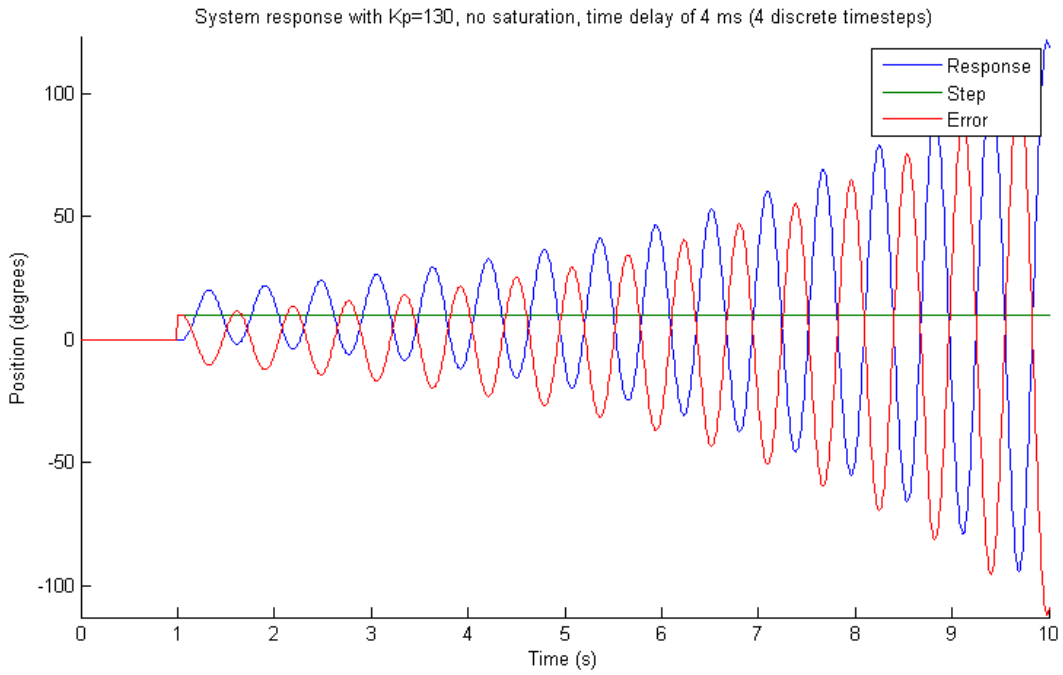


Figure 6: Step response with Kp=130, with a time delay inserted

0.1.4: "Best" Kp

It's clear that a very low Kp will produce unsatisfactory results, since this results in a correspondingly high steady-state error. A higher Kp will reduce this, but at the cost of overshoot and ringing (these can be seen clearly in Figures 3 and 4). If our controller saturates, we get much greater steady-state error, while (undesirably) maintaining some overshoot in the process.

A higher Kp also is (generally) less desirable when time delays are taken into account, though realistically a great enough time delay will drive our system unstable (barring a more complex control loop) regardless of the Kp we choose.

There is no "best" Kp -- it must be chosen to take into account the intended behavior of the system. In one system, high settling time and steady-state error might be acceptable, but overshoot can never be tolerated. In another, we might happily take the ringing and 50% overshoot from a high Kp to

achieve a fast rise time. We would have to in each case analyze the requirements, run simulations, and come up with an optimal solution.

0.1.5: Dependence on ω

When changing ω, it is clear that it greatly affects the system. With ω = 0, there is a double pole at the origin, which is very hard, if not impossible to compensate for with a purely proportional controller. A PI or PID type controller would much better compensate for such a condition. It can be seen from the figures that the system is completely unstable in this configuration. Interestingly, the proportional gain controls what frequency the system oscillates at. With ω = 10, the response time is much faster than the original ω, and can accommodate a higher proportional gain. In fact, a higher proportional gain is required in order to reduce the steady state error. This makes sense, as the system now has a larger bandwidth. However, the real system may not be able to support the higher bandwidth due to issues such as saturation.

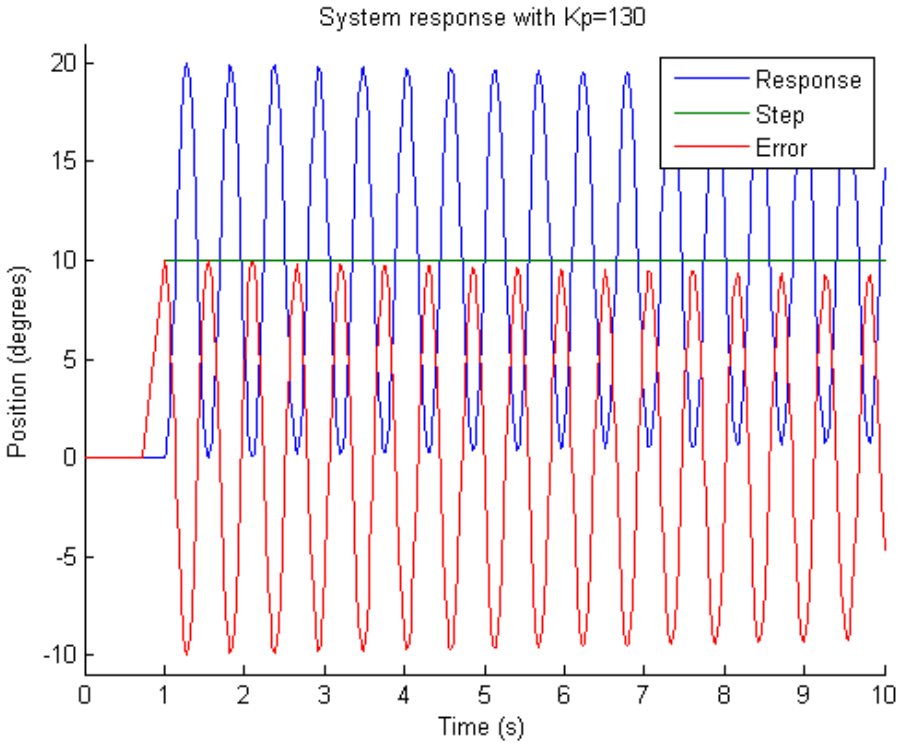


Figure 7: System response with Kp=130 and ω=0

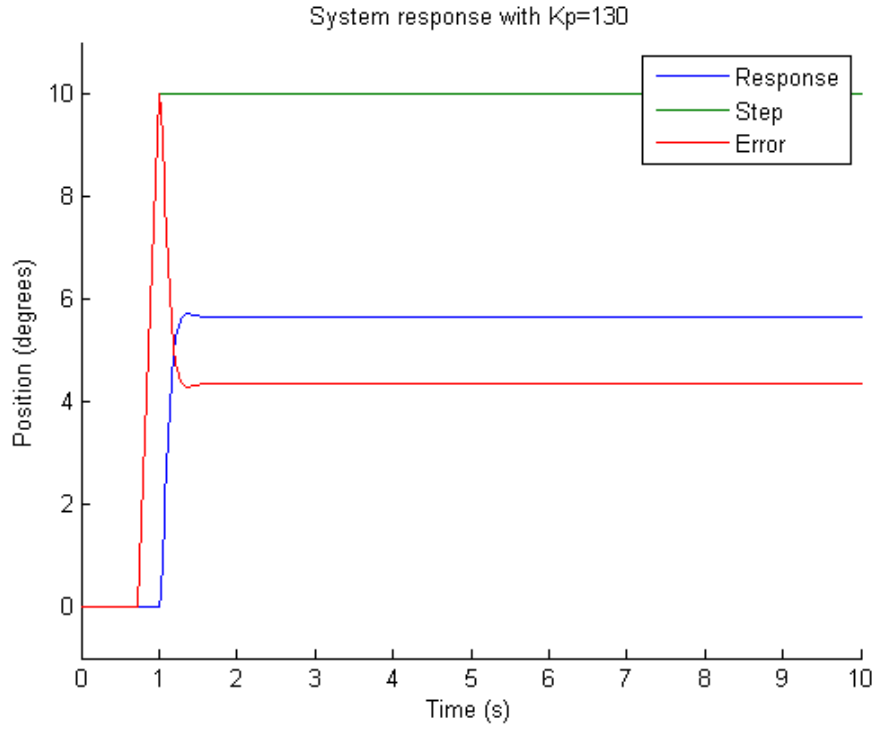


Figure 8: System response with Kp=130 and ω=10